A new iterative method to calculate π

Peter Dion
Department of Anesthesia,
St. Catharines General Hospital, Ontario, Canada
<sleepdoc.awning@sympatico.ca>

Anthony Ho
Department of Anesthesia and Intensive Care
Prince of Wales Hospital, Hong Kong
<hoamh@yahoo.com>

For at least 2000 years people have been trying to calculate the value of π, the ratio of the circumference to the diameter of a circle. We know that π is an irrational number; its decimal representation goes on forever. Early methods were geometric, involving the use of inscribed and circumscribed polygons of a circle (see http://en.wikipedia.org/wiki/Approximations_of_%CF%80#Early_history). However, real accuracy did not come until the use of infinite series techniques, in which one can, by calculating more and more terms, obtain smaller and smaller corrections all leading to a precise value. Such series go on forever, so the limitation on accuracy is how much time one is willing to devote to the task and how fast the computer is, but mainly how quickly your series converges.

One of the first series invented (see http://en.wikipedia.org/wiki/Pi) was by James Gregory (1638–75):

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \cdots$$

As you add up terms in this series it approaches the value of $\frac{\pi}{4}$, that is, it converges, but it does so very slowly. It takes 626 terms just to get π narrowed down to 3.14, that is, two decimal places. The 626th term is $\frac{1}{1251}$, or about one one-thousandth, which corresponds to just the third significant digit in the answer. Isaac Newton used the binomial theorem for real numbers to get a better series (see http://egyptonline.tripod.com/newton.htm).

A still better formula by John Machin (see http://milan.milanovic.org/math/english/pi/machin.html), developed in 1706, enabled hundreds of digits to be calculated:

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}$$

This is not strictly an infinite series, as it just has two terms. However, to calculate each of these terms an infinite series is required.

Not all methods to calculate π are practical. In the so-called "Count Buffon Problem" (see http://www.worsleyschool.net/science/files/buffon/buffon.html), one finds that when one throws toothpicks of length L on a paper ruled with parallel lines spaced a distance L apart, the probability of any toothpick

hitting a line is $\frac{\pi}{2}$. It is an interesting problem and one could actually estimate π by throwing a toothpick repeatedly on a paper. But in practice such an approach for accurate work would be ridiculous.

In this paper we will take a different approach, in which a computer is also used. We will use an iterative approach. A guess at π is used to generate a better estimate, which in turn is used to obtain a still better estimate. This method will initially appear to be very impressive, providing more than sixteen decimal places with only three iterations. However we will see that eventually the calculations become almost unimaginably unwieldy, and that the approach is 'ridiculous' for a serious application. Nevertheless, it makes for a very interesting exercise.

This article will be addressing first year undergraduate university students who will encounter in their mathematics courses the subjects of differentiation, Taylor series, exponentials and natural logarithms, and the idea of array storage in basic computer programming. In terms of teaching concepts we describe the Newton-Raphson method for solving equations in detail as a very useful and lesser known application of differentiation, and then apply it in an unexpected manner to solve a seemingly unrelated problem, the determination of the value of π, thus introducing a new method of approaching one of the most famous endeavours in mathematics. In pursuing this we delve into practical problems in using power series to calculate functions and also into in certain aspects of numerical analysis. This latter topic utilises the Stirling approximation for $n!$, which is introduced but not derived.

We begin with a function $f(x)$. We want to solve the equation $f(x) = 0$. If $f(x)$ is linear or quadratic we can use routine steps to isolate x on one side of the equation. But if $f(x)$ is complicated and we cannot isolate x, there is a technique for a computer to quickly give us a solution provided $f(x)$ is a differentiable function, and it is called the Newton-Raphson method.

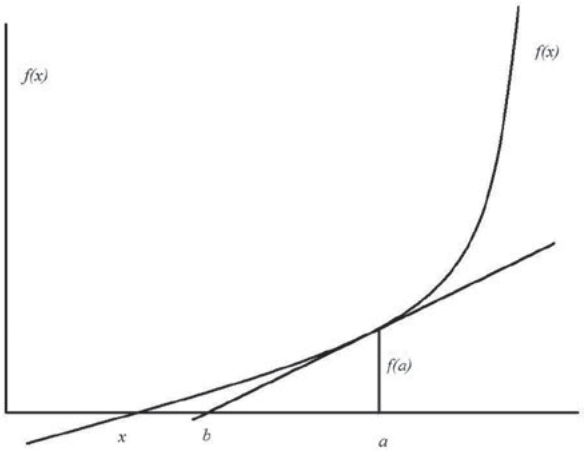

Figure 1. The Newton-Raphson method to find the roots of a function f(x).

In Figure 1 we see some function $f(x)$ which crosses the x-axis at some unknown value x which we want to determine. Our first step is to make a guess somewhere near x using whatever intuition or luck we can muster. We call our guess a, shown on the diagram. Now we know what the function $f(x)$ is, and we can calculate $f(a)$, the height from the x axis to $f(x)$ at point $x = a$. We can also

calculate the derivative of *f(x)*, evaluate it at point *a*, and this gives us the slope of the oblique line shown in the diagram, which is tangent to *f(x)* at point *(a, f(a))*. This line extended intersects the *x*-axis at some value *b*. This value *b* is usually a better estimate than the value *(a)* which we started with.

It is easy to calculate *b*. The slope of the line is *f'(a)*, easily calculated, and that is equal to the rise *f(a)* divided by the run *(a – b)*.

So $$f'(a) = \frac{f(a)}{(a-b)}$$

One solves to obtain $$b = a - \frac{f(a)}{f'(a)} \qquad (1)$$

So now we have a new estimate, *b*. Strictly speaking, this method will not work with any function, but for most conventional functions which cleanly intersect the *x*-axis rather than simply tangentially touching it, and provided our guess *a* is sufficiently close to the point of intersection, it works very effectively. The next step is to repeat the process; we plug our new estimate *b* into (1) wherever *a* appears, to get a newer, better estimate. This process is repeated, and the method usually converges very quickly. One repeats until the new estimate coming out of the formula is the same as the estimate put in, limited by the accuracy (number of decimal places) of your computer. Later we will also use the Newton-Raphson method to solve a fairly complicated equation.

Now, getting back to π, what we need to have is some function *f(x)* which we know is equal to zero for $x = \pi$. If we have that, we can guess a solution, say $x = 3$, and use the Newton-Raphson method to hone in on the real solution, π. Well, sin *x* is a periodic function that produces a nice wavy curve which happens to cross the *x*-axis at $x = \pi$, so we are almost there. Our *f(x)* is sin *x*, the derivative is cos *x*, and so our recursion formula for guesses is:

$$b = a - \frac{\sin a}{\cos a}$$

where our first guess will be $a = 3$.

The computer program we use is a six-line program. We used one written in BASIC, but we present a 'pseudocode' version which might be understood by readers familiar with other languages.

```
Set x=3
OUTPUT "Initial guess was "; x
FOR i = 1 to 10
Let x = x - sin x / cos x
OUTPUT i , x
NEXT i
```

The first two lines assign a value of 3 for *x*, and print it. Lines 3 and 6 are a "FOR - NEXT" loop, using a dummy variable called *i*, which in essence says, "For every value of *i* from 1 to 10, do the stuff between the 'FOR' line and the 'NEXT' line." So ten times you repeat lines 4 and 5. In line 4 you assign to the value *x* the new value $x - \frac{\sin x}{\cos x}$, in line 5 you print out the value of *i*, which

keeps count of which iteration you are on, and print *x*, which is our current estimate for π.

Here is the output of the program:

```
Initial guess was 3
1    3.142546543074278
2    3.141592653300477
3    3.141592653589793
4    3.141592653589793
      .
      .
      .
10   3.141592653589793
```

Our initial guess was 3, and the Newton-Raphson formula gave us a better estimate #1 of 3.14 plus 13 wrong digits. Estimate #2 provided π correct to 9 decimals; estimate #3 was correct to at least 15 decimal places, but we really do not know how good it was because we are already limited by the accuracy of our computer. Most computers provide eight-digit accuracy for real numbers, and 16 digits if the variable is designated as double precision. The fourth to the tenth estimates were simply a repeat of estimate #3, because we already had π correct to within the accuracy of our computer. For reasonably behaving functions *f(x)* the Newton-Raphson method converges very quickly, often doubling the number of digits for each iteration. That means that our third guess may have really been accurate to about 18 digits, the fourth to 36 digits, etc. So at first glance, this seems to be a fabulous method that generates π to fantastic accuracy very quickly!

So perhaps now the authors of this article should be arranging to get over to Stockholm to pick up their Nobel Prize in mathematics. Well, no. Actually there are several problems. First, the Newton-Raphson method is not our discovery. Second, a Noble Prize in mathematics would never be given simply for doing a calculation; one would have to develop an important original theoretical basis for the calculation. Thirdly, there is no such thing as a Nobel Prize for mathematics (that is another story!). But finally, our method to calculate π is all a bit of an illusion. Actually this method, although interesting, is a really poor way to generate π accurately, and in the rest of this paper we will explain why.

To begin with, we have to talk a bit about managing numbers with many digits. We have seen above that our method initially has provided tremendous accuracy, but we are not sure just how accurate because we are representing our desired value, *x*, as a computer variable, limiting us to 16 digits. Now sixteen digits in real practical life provides fantastic accuracy for pretty well any measurement any human will ever make. But remember here we are playing a game of trying to determine a large number of digits for π, regardless of how impractical the result may be. So if we want more than 16 digits, we will have to represent our number π differently, so it can have hundreds, thousands, or even billions of digits. This is a problem faced by all programmers regardless

of the method used to calculate π. One obvious way to do this might be, for example, to represent π as an array *p(i)* of integers, where $p(0)$ might be 3, and $p(5)$ would be the fifth decimal (which would have the value 9), and $p(23423)$ would be the 23 423th decimal of π, and we do not have not a clue what that is.

This, of course, means that any calculations we do involving our number, now an array, will have to be done via some sort of sub-program we must write that enables us to keep track of and add and multiply numbers that will have perhaps millions of digits. This is an annoying problem, but not a conceptually difficult one. In this paper we will ignore this problem and just continue to work with 16 digit numbers represented by a variable such as *x*, and pretend we are actually doing it on a huge array.

This, however, means that with our iterative method we will not be able to tell the computer to simply calculate "sin *x*" anymore because now "*x*" is an array with a huge number of elements. We are going to have to calculate sin *x* ourselves using simple operations (addition, multiplication, division) that we can perform on our huge array representing the number π. So how do we 'manually' calculate sin *x* and cos *x* for some *x*? The answer is that we do it exactly the way the computer does, with a Taylor series; an infinite set of terms which added together will create sin *x* and cos *x*.

The well known Taylor formulae are:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

and

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots$$

Incidentally, it is actually not obvious, but it turns out that the cosine series can be obtained from the sine series by termwise differentiation, as one might hope. Remember earlier we mentioned an infinite series for $\arctan\frac{1}{239}$? Its Taylor series is

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$$

where here you would set $x = \frac{1}{239}$.

So now we know theoretically how to proceed. We handle our *x* as an array, and we calculate sines and cosines with infinite series. It will be in struggling with the practical details of this that we will discover why our method is 'ridiculous'.

We saw early in the paper an infinite series that converged very slowly. If we want a practical method with high accuracy, we do not want a series that takes 600 terms to give us two decimal places. We need our series to converge rapidly. Now in our current situation, the first annoying thing is that, for example, the $x^8/8!$ term in the cosine series will have something like a 3^8 factor in the numerator, a huge number, because *x* has the value of our current estimate for π, which is about 3. True, all terms with high enough powers of *x* will eventually become small, regardless of the value of *x*, because the factorial factor in the denominator will ultimately dominate. But it is most desirable to reduce the number of terms required for a given accuracy, and for that one would

like x to be smaller than three. If x were less than 1, the powers of x would actually help the series converge.

So it is time to reconsider our strategy. Originally with the Newton-Raphson method we sought a value of x that makes $f(x) = 0$. We are not actually limited to that. Suppose we wanted x such that $f(x) = 29$. We simply define a new function, say $g(x)$, which is just $f(x) - 29$. Thus a zero of $g(x)$ is a value of x which makes $f(x) = 29$.

How does this help? Well, initially we picked $\sin x = 0$ to find $x = \pi$. But we can do better. We know that $\sin \frac{\pi}{6} = 0.5$, so we can rewrite this as $f(x) = 0$ where $f(x) = \sin x - 0.5$ and use the Newton-Raphson method to find the value of x where this function is zero. Now our guess for x, and subsequent estimates for x, will be $\frac{\pi}{6}$ or roughly $\frac{1}{2}$. So if we use this value of x, our $\frac{x^8}{8!}$ term will have a factor of $(\frac{1}{2})^8$ instead of 3^8 and things will go much better. Now one might ask, why not then pick $\sin \frac{\pi}{1000}$ or something, and get a really small value of x? The problem is that we have to know the value of $\sin \frac{\pi}{1000}$ *exactly*, as we did above for $\sin \pi$ and $\sin \frac{\pi}{6}$. This is because, otherwise, in essence, we would not know the value of anything exactly. If say, the value of $\sin \frac{\pi}{1000}$ were q, then we would be going in circles, using $\frac{\pi}{1000}$ to find q and using q to find $\frac{\pi}{1000}$. In fact, q would likely be irrational and we would need its value to a billion digits if we wanted to get π to the same number of digits. So we stick with $\frac{\pi}{6}$. So we are now solving $\sin x - 0.5 = 0$, and the derivative of the left side is $\cos x$.

Our Newton-Raphson iteration is now

$$x = x - \frac{\sin x - 0.5}{\cos x}$$

where our initial guess for x is $\frac{1}{2}$.

We are going to do a lot better because the first ten terms of the series $\frac{x^n}{n!}$ with $x = 3$ are 3, 4.5, 4.5, 3.375, 2.025, 1.0125, 0.4339286, 0.1627232, 0.05424107 and 0.01627232, but with $x = 0.5$ they are 0.5, 0.125, 0.0208333, 0.002604, 0.0002604, 0.0000217, 0.00000155, 0.0000000968, 0.00000000538, and 0.000000000269. Consider the tenth term in each. For $x = 3$ it is about 0.016 and will contribute to π from the second decimal place on, but for $x = 0.5$ it is 0.000000000269 and will affect only the tenth decimal place and further, so we will achieve more accuracy faster with the $\frac{\pi}{6}$ series.

So let us do the calculation. It will be instructive first to include terms in our sine and cosine series only up to x^5. Our results are:

Initial guess $\frac{\pi}{6} = 0.5$ or $\pi = 3$
#1 3.140652818991098
#2 3.141577837945346
#3 3.141577879077594
...

(Note: Our series solves for $\frac{\pi}{6}$ but we have displayed the corresponding value of π which is more recognisable.)

Subsequent estimates are all the same as #3. The fifteen decimal digits we have will not change, and we are stuck with a value of π good to only four decimal places. It never gets any better. This is because by including terms only up to those in x^5 we were not really calculating *sin x* or *cos x*, but rather some

functions that are sort of like them. If we include terms up to that involving x^{13} we get the following:

Initial guess $\frac{\pi}{6} = 0.5$ or $\pi = 3$

#1 3.140666842910904

#2 3.141592612362348

#3 3.141592653589793

Again, by the third iteration we have 15 decimals which will not change, but these correctly match the first fifteen decimals in π. One must understand however, that even with terms reaching to x^{13} we are still stalled at a "wrong" value of π. If we had greater accuracy in our calculations (an array for x), we would see that at some point (perhaps the 18th decimal place) our results have diverged from a more accurate value of π. So it is clear that as our estimate for π gets better, we must include more terms in our sine and cosine series in subsequent iterations. Conversely one need not start initially with a huge number of terms in the sine and cosine series. A few terms will suffice initially, because our estimate for π is not too good at the beginning. There is no point carrying terms representing an accuracy not yet justified by our developing estimate for π.

Thus as our value for π gets better and better we need more and more terms in our series. But how many? One strategy would be, at each iteration, to compare the two previous estimates for π, and look to see for how many digits they agree. This tells us our current state of accuracy because these digits will not change any more.

We may thus know how many *digits* we have achieved to date, but we now ask, how many *terms* in our series does this represent? One way to know would be to test each term as generated. For example, let us say we have at some point 1200 decimal places of accuracy. This corresponds to a quantity of 10^{-1200} in terms of precision. We could have our computer ask itself, as *each* term is calculated, "Is this term I'm now calculating as small as 10^{-1200}? And if so, I will calculate no more on this iteration." Since we will end up with thousands of terms, this is a lot of testing. Is there a better way for the computer to know?

There is. If we have k digits of accuracy, this represents a quantity 10^{-k} in precision, and this should be the size of the final term to include, $\frac{x^n}{n!}$ where x is about 0.5 in value.

Thus
$$10^{-k} = \frac{x^n}{n!} \tag{2}$$

and we theoretically could solve for n to get the number of terms to use for this particular iteration. However, it is actually hard to solve for n here, as the tricky item to deal with is $n!$ We thus resort to a clever formula which gives an approximate value for $n!$, called Stirling's formula (see http://mathworld.wolfram.com/StirlingsApproximation.html), which states that

$$n! \approx n^n e^{-n} \left(2\pi n\right)^{\left(\frac{1}{2}\right)}$$

We will use the logarithmic form:

$$\ln(n!) \approx n \ln(n) - n + 0.5 \ln(2\pi n) \tag{3}$$

and also the logarithmic form of equation (2) which is:

$$-k \ln(10) \approx n \ln(x) - \ln(n!) \tag{4}$$

In equation (4) we substitute the expression for $\ln(n!)$ obtained in (3), and after a little rearranging, and assuming $x = \frac{1}{2}$, recalling that x is our rough estimate of $\frac{\pi}{6}$, we have:

$$k\ln(10) \approx n\ln(n) - n + 0.5\ln(2\pi n) + n\ln(2) \qquad (5)$$

Now this *is* an equation begging to be solved with the Newton-Raphson method! We rearrange this into the form $f(n) = 0$ as follows:

$$n\ln(n) - n + 0.5\ln(2\pi n) + \mathrm{n}\ln(2) - k\ln(10) \approx 0$$

and remember we are solving for n here.

Our iterations become:

$$n = n - \frac{f(n)}{f'(n)}$$

where $f(n) = n\ln(n) - n + 0.5\ln(2\pi n) + n\ln(2) - k\ln(10)$

and $$f'(n) = \ln(n) + \frac{0.5}{n} + \ln(2)$$

obtained by differentiating $f(n)$ with respect to n, recognising that k is just a constant here.

Let us try it out. For $k = 15$ decimals of accuracy, let us (stupidly) guess that $n = 3$ solves equation (5). The result of the Newton-Raphson method is:

Initial guess $n = 3$

#1 18.67333

#2 14.07435

#3 13.89607

#4 13.89574

So it seems that about $n = 13$ to 14 terms suffice. Above we found that we got away with 13. So now we know how many terms to use.

But this is all rather cumbersome. Could we not more easily estimate without going into a Newton-Raphson routine with each iteration? Yes we can. In equation (5), some terms can be ignored. The $0.5\ln(2\pi n)$ term becomes small compared to n, which might be in the thousands.

So we have $k\ln(10) \approx n\ln(n) - n + n\ln(2)$. Pulling out a factor of n on the right we have

$k\ln(10) = n\,(\ln(\mathrm{n}) - 1 + \ln(2))$. As we get into huge numbers of digits, the number n of terms will get into the thousands, and $\ln(n)$ will become large compared to 1 or to $\ln(2)$, so we can ignore these and simply have $k\ln(10) \approx n\ln(n)$. If we take this all to the e^{th} power and then take log to base 10 we will get:

$$k \approx n\log_{10}(n) \qquad (6)$$

Recall than n is the n in the term $\frac{x^n}{n!}$ which is the magnitude of a typical digit in the k^{th} decimal place in our estimate for π. We now want to solve equation (6) for n, but this is *still* impossible without the Newton-Raphson method, which we are trying to bypass, so let us just do a bit of trial and error guessing.

For a value of k say, 100 000 let us guess that n were also 100 000. Then we see that $\log(n)$ would be five, so we see we should really make $n = 20\,000$ to satisfy the equation. But then the log of n would change again, but not by much. It would go from 5 to 4.3 and leave the equation still approximately

true. So we can say that n is almost equal to k divided by the log of n (which is almost the same as the log of k), so finally for k digits we just say we need $n \approx \frac{k}{\log(k)}$ terms. So if k is a billion digits, $\log(k)$ is 9, and we need roughly $n = 110\ 000\ 000$ terms in our series expansions for sin x and cosine x. Recall that all these terms have to be calculated, added up to give the sine and cosine values, then used in the Newton-Raphson iteration formula to get our next estimate for π.

Now as a final practical matter, there is one short cut. When one actually does the calculations, one need not work out each term from scratch. For example, if one has the $\frac{x^{16}}{16!}$ term, the $\frac{x^{17}}{17!}$ term is obtained merely by multiplying it by $\frac{x}{17}$. But this little dodge is just hopelessly inadequate to save the day.

By now the reader should correctly be getting very, *very* worried. What we have to do get beyond a billion or so digits in our estimate for π is to calculate about 110 000 000 terms in x (each of which involves raising x to some power from 1 to about 110 000 000), and where x itself is an array with of about a billion digits in it, just to get another iteration.

So by now, you see, our wonderful method has fallen apart. The basic problem is that *no* iterative approach will ever be successful in calculating a large number of digits for π, because having to repeatedly deal with all of the digits again and again eventually becomes intractable. In contrast, in using direct series methods for calculating π as noted early in this paper one does not have to keep dealing with the *whole number for* π. Rather, as one gets into billions of digits, one need work only on calculations dealing with tiny corrections, not with π as a whole. The Newton-Raphson and Taylor power series methods are very effective tools as we have seen, but our use of this particular combination of the two, although initially impressive, has turned out to be a very flawed approach.

So in the end it is perhaps a good thing that there is no Nobel Prize in mathematics, because your authors are certainly not going to be winning it with this kind of stuff. But what is amazing is that now, methods have been developed which allow, for example, the 60 billionth digit of π to be determined without having to figure out all the digits that lead up to it. But of course, this is the stuff of mathematicians with vastly more skill than your authors. In this paper we have just had some fun, nibbling about on the edges of a wonderful problem that has intrigued at least a hundred generations of mathematicians.